# Compromis temps-mémoire
# Time-Memory TradeOff

Cours MPRI niveau 1

**Initiation à la cryptologie**

21 avril 2005

Louis Granboulan

# Plan du cours

1. Introduction
   a) Bibliographie
   b) Introduction et application
2. Quelques TMTO
   a) Stream ciphers
   b) Block ciphers
3. Fonctions aléatoires
   a) Permutations, cycles
   b) Fonctions
4. Améliorations
   a) Biryukov-Shamir
   b) Rivest et Oechslin
   c) Fiat-Naor
5. Autre : baby-step/giant-step, de Shanks

# Bibliographie


- Rediscovery of Time Memory Tradeoffs
  - Jin Hong & Palash Sarkar, 2005
  - http://eprint.iacr.org/2005/090/
  - Une synthèse de l'état de l'art
- Making a faster cryptanalytic Time-Memory Trade-Off
  - Philippe Oechslin, 2003
  - http://lasecwww.epfl.ch/php_code/publications/search.php?ref=Oech03
  - Description des Rainbow Tables
- A cryptanalytic time-memory trade off
  - M. E. Hellman, 1980
  - IEEE Transactions on Information Theory
  - Article fondateur
- Rigorous time/space tradeoffs for inverting functions
  - Amos Fiat & Moni Naor, 1991
  - STOC 1991
  - Une étude rigoureuse de la technique de Hellman

# Introduction

- Cryptanalyse avec précalculs
  - principalement dans les cas où la meilleure attaque est la recherche exhaustive.

- Les paramètres de complexité d'une attaque :
  - P temps du précalcul
  - M mémoire pour stocker le résultat du précalcul
  - T temps de cryptanalyse
  - N temps de la recherche exhaustive

# Modèles de calcul

- Si l'attaquant est une machine de Turing
  - Déplacement sur une bande ⇒ M≤T
  - Se protéger contre tout attaquant : P non borné
  - Donc les modèles formels de sécurité ne peuvent contraindre P
- Réalisme
  - L'attaquant est une machine du monde réel
  - P est le temps de construction de l'attaquant, qui doit donc être limité par le paramètre de sécurité

# Idée naïve

- Nouvelle notation : $\tilde{O}(x)$ pour négliger les facteurs polylogarithmiques

- On précalcule toute la recherche exhaustive qu'on stocke dans la table.
  - $P=M=O(N)$ ; $T=\tilde{O}(1)$
- À comparer avec la recherche exhaustive
  - $P=M=O(1)$ ; $T=O(N)$
- Objectif
  - Minimiser M et T ; $P=\tilde{O}(N)$ n'est pas un problème

# Application : calcul de préimage

- Une fonction à sens unique f, dont on veut calculer rapidement des préimages

- Exemple : hachage
  - Pour retrouver des mots de passe à partir du haché
- Exemple : chiffrement
  - Pour retrouver la clef à partir du chiffré d'un clair connu et prédéterminé

# TMTO de Babbage et Golic

- Proposé pour des stream ciphers (1995)
  - Cette attaque permet d'affirmer que l'état interne d'un système de génération de flot doit avoir une taille double du paramètre de sécurité.
- Générateur de flot
  - Un état interne x, mis à jour par $x_{i+1}=u(x_i)$
  - Le flot est fabriqué par $s_i=o(x_i)$
  - Nombre d'états internes : N
  - La fonction u est une permutation de X avec un cycle de taille N
- Efficacité en pratique :
  - $P=M=\tilde{O}(N^{1/2})$ ; $T=\tilde{O}(N^{1/2})$
  - Taille de flot nécessaire D=T

# TMTO de Babbage et Golic

- On précalcule m débuts de flots
  - On part d'un élément aléatoire $x=x_0$ et on calcule $s_i=o(x_i)$ et $x_{i+1}=u(x_i)$
  - Appelons f la fonction qui à x associe la séquence de log N bits $[s_0 \ldots s_{l-1}]$.
  - On stocke donc des paires (x, f(x) )
- Attaque
  - On utilise un flot de taille d, dans lequel on cherche l'une des m séquences, ce qui prend un temps $t=\tilde{O}(d)$.
  - La probabilité de succès est raisonnable (paradoxe des anniversaires) si $md=N$, par exemple $m=t=d=N^{1/2}$

# TMTO de Hellman

- Proposé pour le DES
- Cette attaque (et ses améliorations) sont utilisées pour retrouver les mots de passe Windows (LanManager Hash)

- Efficacité en pratique :
  - $P=\tilde{O}(N)$ ; $T=M=\tilde{O}(N^{2/3})$

# TMTO de Hellman

- Fonction f : X → X
  - Supposée aléatoire ; par exemple $f(x)=DES_x(0)$
- On précalcule m chaînes de longueur t
  - Une chaîne part d'un élément aléatoire $x_0$ et calcule $x_{i+1}=f(x_i)$, on stocke $(x_0,x_t)$
  - Complexité : mt
- Attaque
  - Pour trouver une préimage de x, on itère f jusqu'à tomber sur l'un des $x_t$ ; puis on itère f à partir du $x_0$ correspondant
  - Problème : les collisions

# TMTO de Hellman

- On choisit t permutations $g_i : X \rightarrow X$
  - On fait ainsi t fois le précalcul précédent, en utilisant $g_i \circ f$ au lieu de f
  - On précalcule donc t m chaînes de longueur t
    - Une chaîne part d'un élément aléatoire $x_0$ et calcule $x_{i+1}=g_i \circ f(x_i)$, on stocke $(x_0,x_t)$
- Attaque
  - Complexité : $M=tm$, $T=t^2$
  - On choisit des paramètres tels que $t^2m=N$
  - $t=m=\tilde{O}(N^{1/3})$
  - Heuristique : tous les éléments de X ont été parcourus par le précalcul

# Permutations aléatoires

- Une permutation d'un ensemble de N éléments peut être décomposée en cycles
- La taille moyenne pondérée des cycles est (N+1)/2
  - Pour x fixé, on compte les permutations telles que l'orbite de x soit de taille k : elles sont (N−1)!
  - Donc la taille moyenne pondérée est $\sum$ k/N
- La taille moyenne du plus grand cycle est environ 0.6 N

# Fonctions aléatoires

- Décomposition
  - Une fonction peut être décomposée en soleils (cycle où arrivent des arbres)
- Aléatoire
  - Une fonction aléatoire contient presque uniquement un gros soleil, donc le cycle est de taille $N^{1/2}$ et les arbres qui y sont accrochés sont de hauteur $N^{1/2}$
- TMTO
  - Si on note I(x) le degré entrant (nombre de préimages de x) et $q=(\sum I(x)^2)/N^2$ la probabilité de collision, alors le TMTO de Hellman s'applique si $q=\tilde{O}(1/N)$

# Biryukov-Shamir

- Principe
  - Idée de Hellman (permutations $g_i$) adaptée aux stream ciphers
  - Gain : flot nécessaire $D=\tilde{O}(N^{1/4})$ au lieu de $\tilde{O}(N^{1/2})$
- Détails
  - Au lieu de calculer t tables de m chaînes de longueur t, on n'en calcule que t/D.
  - Pour $t^2m=N$, cela couvre une proportion 1/D des états, qui entre donc en collision avec l'un des D flots
  - Complexité : $M=tm/D$, $T=t^2$
  - On choisit $t=\tilde{O}(N^{1/4})$, $m=\tilde{O}(N^{1/2})$, pour $D=\tilde{O}(N^{1/4})$

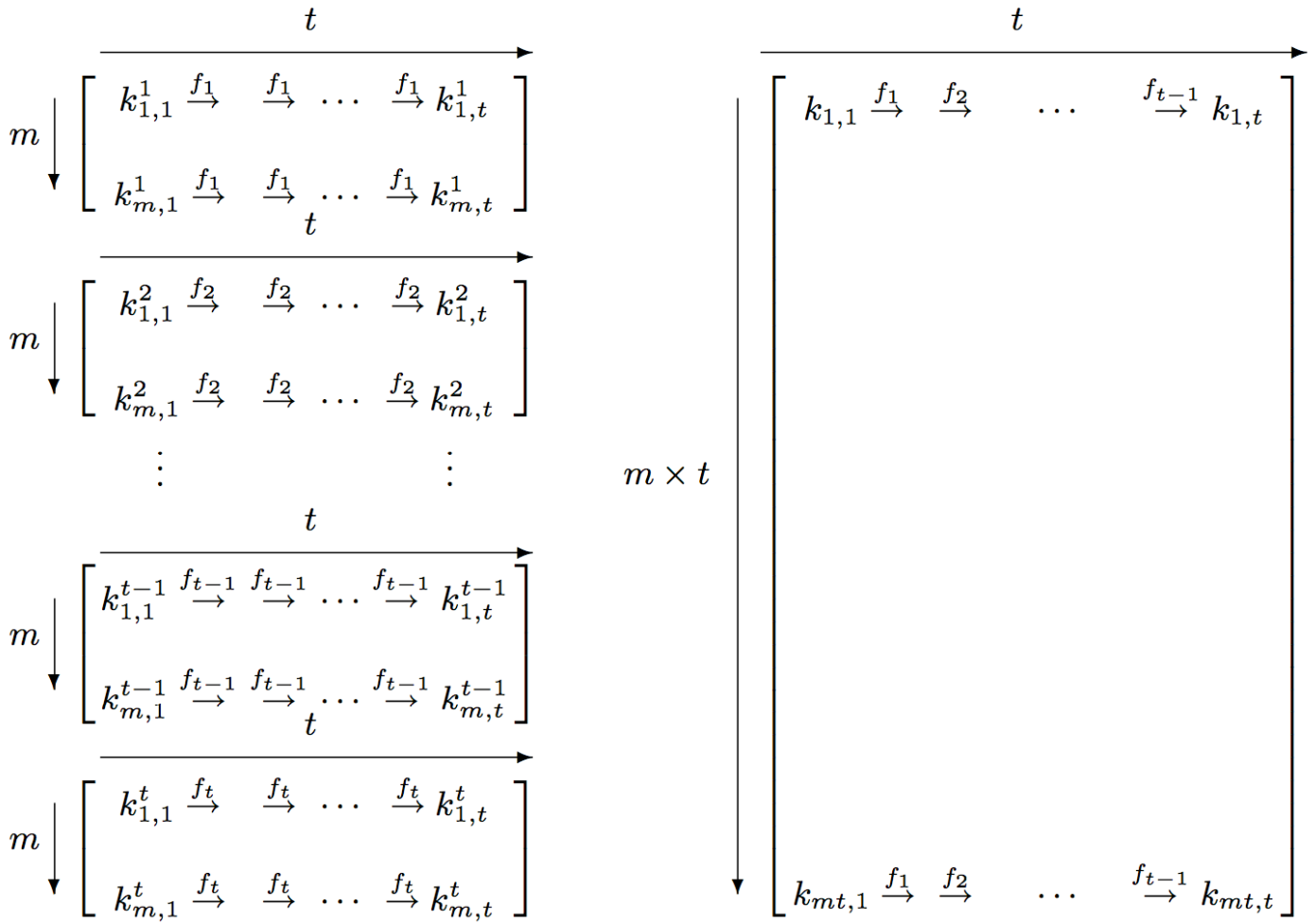
# Rivest

- Utilisation de points distingués
  - Amélioration du TMTO de Hellman
  - Au lieu calculer des chaînes de longueur t, on calcule des chaînes jusqu'à arriver à une valeur ayant des propriétés particulières.
- Avantages
  - Évite de perdre du temps dans les petites boucles.
  - Collisions plus facilement détectées, car ont la même fin.

# Oechslin

- Utilisation de tables arc-en-ciel
  - Autre amélioration du TMTO de Hellman
  - On change $g_i$ à chaque appel, lors de la construction d'une chaîne : une unique table de t m chaînes, au lieu de t tables de m chaînes
- Avantages
  - Chaînes de longueur fixe t
  - Collision n'implique pas fusion
  - Gain de temps $T=t^2/2$

# Oechslin : les tables



# Fiat-Naor

- Remarque
  - Le TMTO de Hellman ne marche que si f est aléatoire, pas si f est quelconque
  - Cas pathologique : $N^{1-\varepsilon}$ valeurs ont la même image
- Résultats
  - Le TMTO de Hellman correspond à $TM^2=N^3q(f)$
  - Pour une fonction quelconque, on peut obtenir $TM^3=N^3$, ce qui donne $T=M=\tilde{O}(N^{3/4})$

# Résumé

Fiat-Naor
- fonction quelconque
- Formules $N=MT^{1/3}$ ; $P=N$
- Paramètres $P=N$ ; $M=T=\tilde{O}(N^{3/4})$

Hellman, Rivest ou Oechslin
- fonction aléatoire
- Formules $N=MT^{1/2}$ ; $P=N$
- Paramètres $P=N$ ; $M=T=\tilde{O}(N^{2/3})$
- Application : bloc ou stream avec N clefs possibles

Cas des permutations
- Formules $N=MT$ ; $P=N$
- Paramètres $P=N$ ; $M=T=\tilde{O}(N^{1/2})$

Babbage et Golic
- flot ; N états internes ; D flots
- Formules $N=TM$ ; $P=M$ ; $D=T$
- Paramètres $P=M=T=D=\tilde{O}(N^{1/2})$

Biryukov-Shamir
- flot ; N états internes ; D flots
- Formules $N/D=MT^{1/2}$ ; $P=N/D$
- Paramètres $P=\tilde{O}(N^{3/4})$ ; $M=T=\tilde{O}(N^{1/2})$ ; $D=\tilde{O}(N^{1/4})$

Hong-Sarkar
- flot ; N clefs possibles
- idem ci-dessus, pour retrouver une clef parmi D

# Baby-step/giant-step de Shanks

- Technique de calcul de logarithme discret
  - Étant donné h dans le groupe engendré par g, d'ordre N, trouver x tel que h=x.g
  - Idée, poser M=√N, décomposer x=iM+j
- Précalcul : mémoire et temps √N
  - Posons g'=-M.g
  - Calcul et stockage des m paires (j,j.g)
- Calcul : temps √N log N
  - Collision avec l'une des m valeurs h+i.g'

# Fin